



2/6

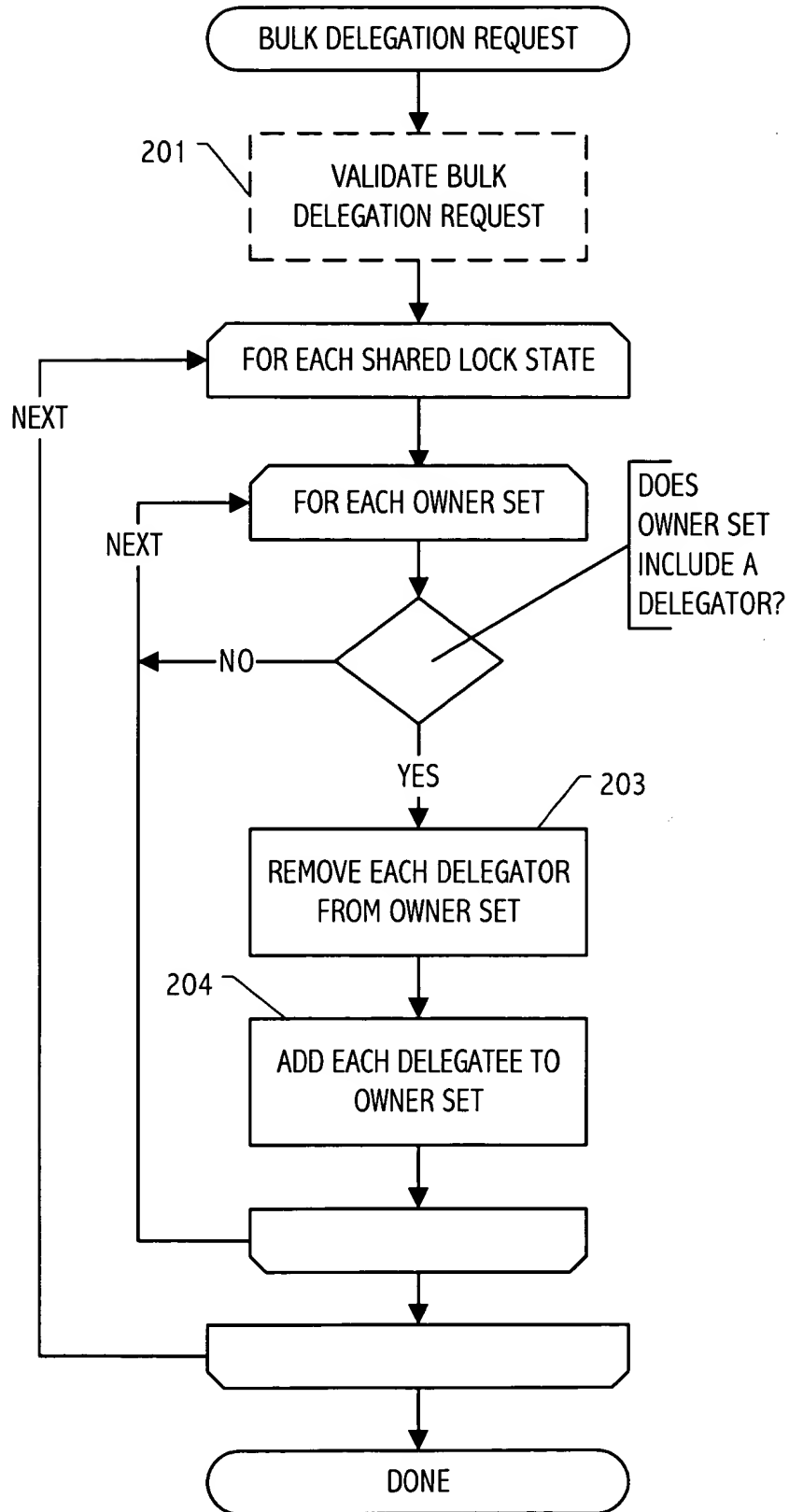


FIG. 2

```

delegate(delegators, delegates)
begin
  foreach l in TSL
    if  $\exists M, (delegators \cap Owners(l, M) \neq \emptyset) \wedge (\nexists M_i, M_i > M \wedge (delegators \cap Owners(l, M_i) \neq \emptyset))$ 
      TSL.remove(l)
      // modify its owner set to reflect the effect of delegation
      foreach  $M_i, M_i \leq M$ 
         $Owners(l, M_i) \leftarrow [Owners(l, M_i) - delegators] \cup delegates$ 
      end
      // does the new value duplicate an existing shared lock state ?
      if TSL.contains(l)
        // yes. record the "original" being duplicated
        // and add the shared lock state to the set of duplicates.
         $original(l) \leftarrow TSL.get(l)$ 
        duplicates.add(l)
      else
        // no. Re-enter the modified shared lock in the TSL.
        TSL.add(l)
      endif
    endif
  end
  // Process duplicates now.
  foreach l in duplicates
    if  $\exists M, (delegators \cap Owners(l, M) \neq \emptyset) \wedge (\nexists M_i, M_i > M \wedge (delegators \cap Owners(l, M_i) \neq \emptyset))$ 
      // modify its owner set to reflect the effect of delegation
      foreach  $M_i, M_i \leq M$ 
         $Owners(l, M_i) \leftarrow [Owners(l, M_i) - delegators] \cup delegates$ 
      end
    endif
  end
end
end
end

```

FIG. 3

4/6

```

// Determine the validity of a delegating a lock set to the value l
boolean isValid(delegates, delegates, l)
begin
    if Owners(l, Write) =  $\emptyset$ 
        return true
    endif
    // at least one write lock owner
    if Owners(l, Write)  $\cap$  delegates =  $\emptyset$ 
        // All the delegates are read owners.
        // The delegation is valid if all delegates can ignore read-write
        // conflicts with the write owners.
        return  $\forall t \in \text{delegates}, \text{Owners}(l, \text{Write}) \subseteq \text{ICW}(t, \text{rw})$ 
    endif
    // the lock is delegated in write mode - all delegates must ignore
    // write-write conflicts between each others and with each remaining
    // owners of the lock in write mode. Also, write-read conflicts should
    // be ignored with remaining owners of the lock in read mode.
    if |delegates| > 1
        // More than one delegatee
        if  $\exists t \in \text{delegates}, \exists c \in \{\text{rw}, \text{wr}, \text{ww}\}, \text{delegates} \not\subseteq \text{ICW}(t, c)$ 
            return false
        endif
    endif
    if  $\exists t \in \text{delegates}, (\text{Owners}(l, \text{Write}) - \text{delegates}) \not\subseteq \text{ICW}(t, \text{ww})$ 
        return false
    endif
    if  $\exists t \in \text{delegates}, (\text{Owners}(l, \text{Read}) - \text{delegates}) \not\subseteq \text{ICW}(t, \text{wr})$ 
        return false
    endif
    return true
end

```

FIG. 4

5/6

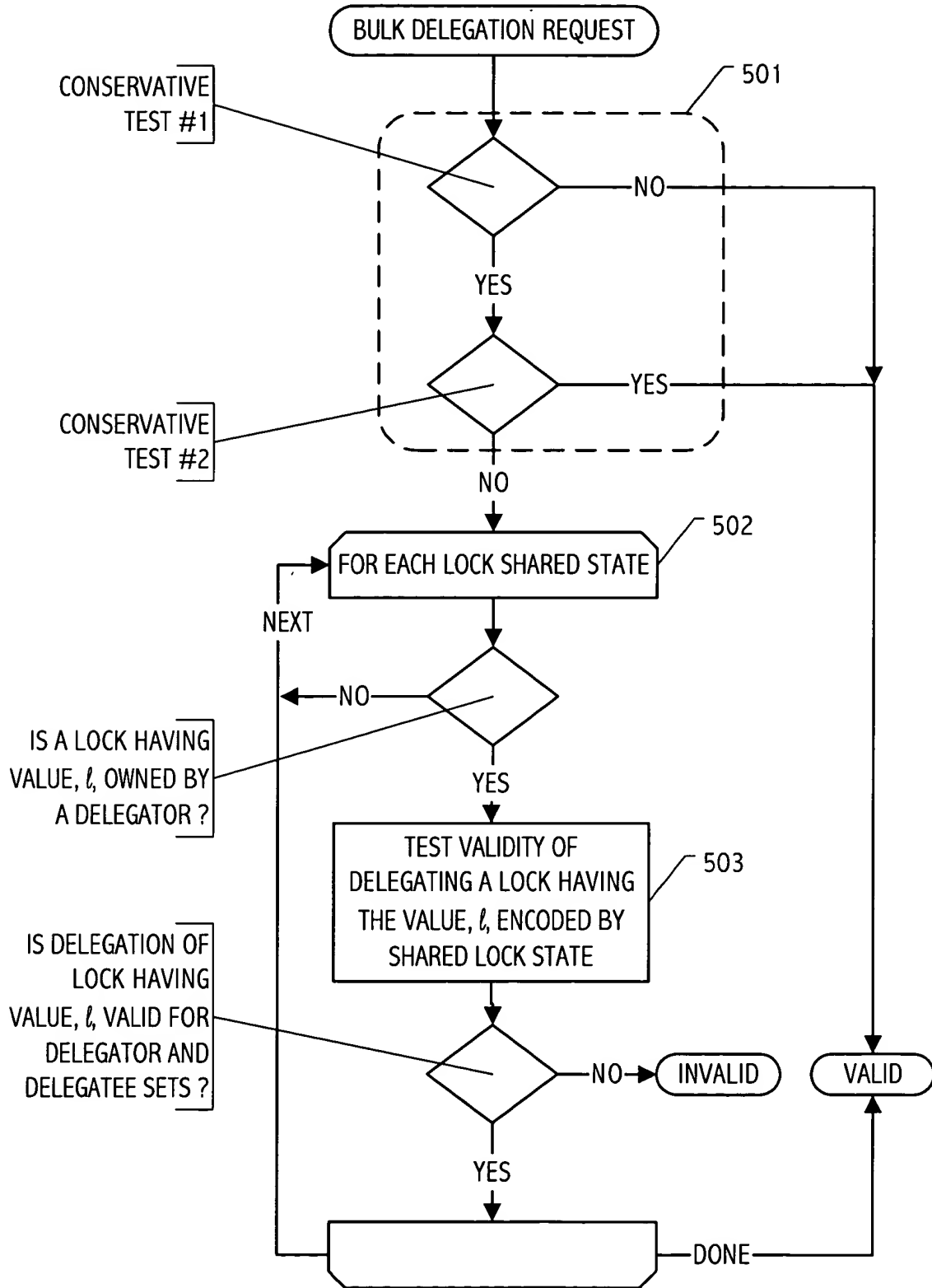


FIG. 5

6/6

```

// Determine the validity of a bulk lock delegation
boolean isValid(delegates, delegates)
begin
602 — if  $wset \cap delegates = \emptyset$ 
    return true
603 — else if ( $\forall t_d \in delegates, \forall t_s \in delegates, \forall C \in \{rw, wr, ww\},$ 
    ( $ICW(t_s, C) - (delegates \cup \{t_d\}) \subseteq ICW(t_d, C)$ )
    // if at least one of the delegated lock is a write lock, the request
    // is valid only if the delegates can ignore all conflicts which each other
    if  $\exists l, Owners(l, Write) \cap delegates \neq \emptyset$ 
        return ( $\forall t \in delegates, \forall c \in \{rw, wr, ww\}, delegates \subseteq ICW(t, c)$ )
    else
        return true
    endif
else
    // The two conservative tests have failed
    foreach l in TSLS
        if  $Owners(l, W) \neq \emptyset \wedge (\exists M, Owners(l, M) \cap delegates \neq \emptyset)$ 
601 — if  $\neg isValid(delegates, delegates, l)$ 
            return false
        endif
    endif
end
endif
return true
end

```

FIG. 6